An Efficient Method of Data Inconsistency Check for Very Large Relations

Hyontai Sug,

Dongseo University, Busan, South Korea

Summary
In order to deal with the data inconsistency problem in very large relations, a new method is suggested to help users find inconsistent data conveniently based on the functional dependencies of the corresponding relation variables. The possible wrong data are found by applying an association rule finding algorithm in a limited way. An experiment showed very promising result.

Key words:
Data inconsistency check, very large relations.

1. Introduction

As computers become very popular nowadays, lots of databases are constructed and used because of high availability of database management systems without having to pay much to acquire the systems. For example, one of very widely used database management systems, MySQL can be freely used unless the system is used for commercial purpose [1]. Another very popular database management system for relatively small databases, Microsoft ACCESS [2] is very widely used because of good user interfaces that enable users to construct tables and make queries easily. Even Microsoft SQL server [3] for middle-sized databases is relatively very easy to use because of its similarity in user interfaces with those of Microsoft ACCESS. This ease of use of relational database management systems has contributed to let many people who do not have exact knowledge on normal forms of relation variables design relational databases with ease. But, this ease of use might play a role of some bad point in the respect of data management. Because relational databases are consisted of relations that resemble conventional tables, users or designers of the databases consider that the structure of databases is very simple so that they do not pay much attention to the structures of the relations. They might consider the relations are just like conventional tables and they want to store data in a small number of tables as much as possible, because they usually think that the complexity of making queries from the tables is increased as the number of tables is increased, as the author experienced from many people. Therefore, it is highly possible that the relations might contain some redundant information due to the small number of tables. For example, consider the following example database for a video DVD rental shop. The shop's database has a table called rentalTable to store the DVD rental information. The table has an attribute set like {*renter_number*, renter_name, renter_address, *DVD_number*, *rent_date*, return_date} where words in slanted characters represent the attributes' role as a primary key. And the designer of the database does not want to have a separate table to store the renter's information, because most of the customers are one-time customers, and reports to be printed out need all the information in the rentalTable. Moreover, separate table structures like rental_info{*DVD_number, rent_date*, return_date} and renter_info{*renter_number*, renter_name, renter_address} make inputting data and printing reports from the two tables harder. But due to the structure of rentalTable, if some customers rent more than once, redundant information like renter_name, renter_address can exist in the table. This redundancy may cause data inconsistency problem, if the redundant information has not been updated unanimously. The problem become worse when the structure of databases is more complex and the number of tables is increased, which is very common in real world databases.

One may suggest to using a sorting method to find out inconsistent data. But, manual checking after sorting is very difficult when the number of tuples is very large and the domains of corresponding attributes are large. If we assume that we want to check inconsistent data between two attributes and each attribute has k and q number of values, the possible number of combination of attribute values is kq. So, if k and q are large numbers, manual checking for data inconsistency will be very difficult. This paper suggests a consistency checking method to deal with such situation. We suggest a method to solve the problems of data inconsistency based on an approach developed from association rules and functional dependencies between attributes.

We will first discuss related works in section 2, in section 3 we present our method in detail and in section 4 we illustrate our method through experimentation. Finally in section 5, we present conclusions.

Manuscript received October 5th, 2007.
Manuscript reviced October 20th, 2007.

2. Related work

Because we want to apply association rule discovery algorithms in a limited way, let's take a look at the algorithm briefly. Association rules are rules about how often sets of items occur together. These rules produce information on patterns or regularities that exist in a database. For example, suppose we have collected a set of transactional data in a supermarket for some period of time. We might find an association rule from the data stating `instant coffee => non-dairy creamer (80%),' which means that 80% of customers who buy instant coffee also buy non-dairy creamer. Such rules can be useful for decision making on sales, for example, determining an appropriate layout for items in the store. Basically, association rule finding algorithms search exhaustively to find association patterns. So, there can be many association rules and much computing time can be needed for large target databases, because of the exhaustive nature of the algorithm. So, supplying appropriate minimum support based on the target database size is important.

There are many good algorithms to find association patterns efficiently; a standard association algorithm, Apriori, a large main memory-based algorithm like AprioriTid [4], the hash-table based algorithm, DHP [5], random sample based algorithms [6], tree structure-based algorithm [7], and a parallel version of the algorithm [8]. ART [9] tried to achieve scalability by using decision list, but the exhaustive nature of the algorithm makes a limit to the applicability of the algorithm.

3. Proposing method

The following is a formal definition of the problem. Let I = $\{i_1, i_2, ..., i_n\}$ be a set of items that are consisted of attribute-value pairs with attributes of an interested functional dependency in a relation R. Each tuple in R has a unique primary key. For an itemset $X \subset I$,

Support_Number(X) = the number of tuples in R containing X.

An association rule is an implication of the form Y => Z where each Y and Z has different attributes, $Y \subset I$, $Z \subset I$, and $Y \cap Z \neq \phi$. The consistency of stored attribute values in a relation is represented with confidences of found association rules. The confidence C% of the association rule Y => Z implies that among all the tuples which contain itemset Y, C% of them also contains itemset Z. So, the confidence of a association rule Y => Z is computed as the ratio:

Support_Number($Y \wedge Z$)/Support_Number(Y).

Note that $Y \wedge Z$ means items in a tuple. If there is a functional dependency between attributes of Y and Z, the confidence value should be 100%. On the other hand, even though there is a functional dependency between attributes of Y and Z, if the confidence of association rule Y => Z is less than 100%, then there should be inconsistency in data. For example, if there is a functional dependency between attributes A and B, and there are two tuples where one tuple has values $A=a_1$, $B=b_1$, and the other tuple has values $A=a_1$, $B=b_2$, then the confidence for rule $A=a_1$ => $B=b_1$ or $A=a_1$ => $B=b_2$ is 50% and data is in inconsistency.

The functional dependencies we want to check has the property that each right and left hand side of the functional dependency consists of one attribute. If some functional dependencies have several attributes in their right hand side, we need to separate the attributes of the right hand side one by one. But, the separation doesn't matter, because we can always decompose the right hand side by Armstrong's axiom [10]. For example, the functional dependency $A \rightarrow \{B, C\}$ is equivalent to functional dependencies $A \rightarrow B$ and $A \rightarrow C$.

If each attribute in left hand side and right hand side has k and q number of values, the possible number of rules is kq. So, if k and q are large numbers, the resulting combination of attribute values will be very large also. For example, if k and q are 30, then the possible combination is 900 so that manual inspection will be very difficult.

To find inconsistent data in a given relation we apply the following steps for each user-selected functional dependencies in the relation.

(i) Select a functional dependency (FD) for data inconsistency check.
(ii) Run association rule algorithm for the attributes in the given FD with the parameter of minimum support number of 1.
(iii) Generate rules for the right hand side of the FD.
(iv) Find inconsistent data with association rule with confidence less than 100%.

Note that in (ii) we apply association rule finding algorithm in limited way so that the computing time to find frequent patterns with very small value of minimum support number, 1, cannot cause any significant computing time. We adapt a hash-table based association rule finding algorithm like DHP, because the algorithm can generate short association rules with length two very fast compared to other association rule finding algorithms.

Moreover, in (iii) because we generate rules with fixed right hand side, the number of rules generated is much less than conventional association rule generation algorithms. For example, if the number of attribute values in right hand side is two, we have only half number of rules than conventional association rule algorithms.

4. Experimentation

An experiment was run using a database in UCI machine learning repository [11] called 'census-income' to see the effectiveness of the method. In the original data the number of instances for training is 199,523 in size of 99MB data file. Class probabilities for label -50000 and 50000+ are 93.8% and 6.2% respectively. The database was selected because it is relatively very large and contains lots of manifest facts. The total number of attributes is 41. Among them eight attributes are continuous attributes. We selected two attributes among the 41 attributes for the experiment. We assumed that a functional dependency exists between the selected two attributes, attribute sex and attribute class. The used computer is VAIO notebook computer with 512 MB main memory. It took only a few seconds of computing time. We found the following rules.

- **IF** sex = female **THEN** class = -50000. (freq=101,321, cf=0.97)
- **IF** sex = female **THEN** class = 50000+. (freq=2,663, cf=0.03)
- **IF** sex = male **THEN** class = -50000. (freq=85,820, cf=0.9)
- **IF** sex = male **THEN** class = 50000+. (freq=9,719, cf=0.1)

In the above rules freq and cf mean frequency and confidence respectively. If the functional dependency sex $\rightarrow$ class is true, there should be no contradicting data like the above. Next step will be to correct the inconsistent data.

5. Conclusions

As computers become very popular nowadays, lots of databases are constructed and used because of the high availability of database management systems without having to pay much to get the systems. So, many databases are constructed and lots of data are being stored. But, it is difficult to say all the databases are designed well to meet the constraints of, so called, the normal forms, because it is highly possible that some databases or possibly many databases are not designed with much consideration about normalization due to the fact that not all of the designers of the databases are good designers. So, it is highly possible that some data in the databases are redundant so that the databases may contain inconsistent data.

This paper suggests an effect method to find such inconsistent data based on functional dependencies between attributes in a relation. We apply an association rule algorithm with respect to the attribute sets in the functional dependencies in a limited way to effectively find out the inconsistent data. We showed by experiment that it is very effective in finding such inconsistencies.

References

[1] B. Forta, MySQL Crash Course (Sams Teach Yourself in 10 Minutes), Sams, 2005.
[2] J. Viescas, J. Conrad, Microsoft Office Access™ 2007 Inside Out, Microsoft Press, 2007.
[3] R. Rankins, P. Bertucci, C. Gallelli, A.T. Silverstein, Microsoft® SQL Server 2005 Unleashed, Sams, 2006.
[4] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A.I. Verkamo, "Fast Discovery of Association Rules," In Advances in Knowledge Discovery and Data Mining, U.M. Fayyad, G. Piatetsky-Shapiro, P. Smith, R. Uthurusamy ed., pp.307-328, AAAI Press/The MIT Press, 1996.
[5] J.S. Park, M. Chen, P.S. Yu, "Using a Hash-Based Method with Transaction Trimming for Mining Association Rules," IEEE Transactions on Knowledge and Data Engineering, vol.9, no.5, pp. 813-825, Sep. 1997.
[6] H. Toivonen, Discovery of Frequent Patterns in Large Data Collections, phD thesis, Department of Computer Science, University of Helsinki, Finland, 1996.
[7] J. Han, J. Pei, Y. Yin, "Mining Frequent Patterns without Candidate Generation," *SIGMOD'00*, Dallas, TX, May 2000.
[8] A. Savasere, E. Omiecinski, S. Navathe, An Efficient Algorithm for Mining Association Rules in Large Databases, College of Computing, Georgia Institute of Technology, Technical Report No.: GIT-CC-95-04.
[9] F. Berzal, J. Cubero, D. Sanchez, and J.M. Serrano, "ART: A Hybrid Classification Model," Machine Learning, vol.54, pp.67-92, 2004.
[10] C. J. Date, An Introduction to Database Systems, 8th ed. Addison-Wesley, pp.338-339, 2004.
[11]] S. Hettich, S.D. Bay, The UCI KDD archive, Technical Report, University of California, Irvine, Department of Information and Computer Science, 1999.

Hyontai Sug received the B.S. degree in Computer Science and Statistics from Pusan National University, M.S. degree in Computer Science from Hankuk University of Foreign Studies, and Ph.D. degree in Computer and Information Science and Engineering from University of Florida in 1983, 1986, and 1998 respectively. During 1986-1992, he worked for Agency of Defense Development as a researcher, and during 1999-2001, he was a full-time lecturer of Pusan University of Foreign Studies. He is now with Dongseo University as an associate professor since 2001.